

Звіт
Лабораторна робота 2
ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ
Большаков Андрій МІТ-31
https://github.com/Utilka/OOP_labs_Univ

Тема :
НАСЛІДУВАННЯ ТА ПОЛІМОРФІЗМ У ОБ'ЄКТАХ ПАЙТОН

Мета: ознайомитися з поняттям наслідування в об'єктно-орієнтованому програмуванні, навчитися створювати класи та об'єкти з наслідуванням у Пайтон.

ЗАВДАННЯ 1 (6 балів у разі повного і правильного виконання роботи).
Самостійно опишіть необхідні класи згідно свого варіанту завдання (таблиця 2.2), визначте зв'язки між ними, аргументуйте своє рішення. Продемонструйте функціонування усіх описаних функцій, створивши прикладну програму. Особливу увагу при виконанні завдання зверніть на уникнення дублювання коду!!!

4	<u>Предметна область:</u> описати ієрархію типів вагонів потяга для симуляції роботи залізниці (три типи вагонів з різними рівнями забезпечення комфорту). Кожен вагон при формуванні потяга повинен мати провідника/провідницю, при чому провідник/провідниця можуть працювати щоразу у іншому вагоні при наступному формуванні потяга.
---	--

у нас есть 2 связи

связь между Train и Car агрегация:

при создании поезда проверка того что у всех вагонов одинаковая ширина «колес»

связь между Train, PassengerCar, RailroadsConductor последовательная агрегация:

при отправке поезда дополнительная логика того что в каждом PassengerCar должен быть RailroadsConductor

class Car:

```
def __init__(self, wheel_width, width, height, length):
    self.wheel_width = wheel_width
    self.width = width
    self.height = height
    self.length = length
```

class PassengerCar(Car):

```
def __init__(self, wheel_width, width, height, length, passenger_count, railroads_conductor, ticket_price):
    Car.__init__(self, wheel_width, width, height, length)
    self.passenger_count = passenger_count
    self.railroads_conductor = railroads_conductor
```

```
self.ticket_price = ticket_price
```

```
class CoupeCar(PassengerCar):
```

```
    def __init__(self, railroads_conductor):
        PassengerCar.__init__(self, 100, 120, 360, 6 * 360, 40, railroads_conductor, 50)
```

```
class EconomyCar(PassengerCar):
```

```
    def __init__(self, railroads_conductor):
        PassengerCar.__init__(self, 100, 60, 360, 6 * 360, 100, railroads_conductor, 20)
```

```
class FreightCar(Car):
```

```
    def __init__(self, wheel_width, width, height, length, volume):
        Car.__init__(self, wheel_width, width, height, length)
        self.volume = volume
```

```
class RailroadsConductor:
```

```
    def __init__(self, name):
        self.name = name
```

```
class Train:
```

```
    def __init__(self, name, engine, car_list):
        Train._check_cars(car_list)
        self.car_list = car_list
        self.engine = engine
        self.name = name
```

```
@staticmethod
```

```
def _check_cars(car_list):
    car_target_wheel_width = car_list[0].wheel_width
    for car in car_list:
        if car_target_wheel_width != car.wheel_width:
            raise Exception("Not all cars have same wheel_width")
```

```
def _check_departure_ready(self):
    for car in self.car_list:
        if isinstance(car, PassengerCar):
            if not isinstance(car.railroads_conductor, RailroadsConductor):
                raise Exception("Not all cars have railroads_conductor assigned")
```

```
def depart(self):
    self._check_departure_ready()
    print(f"Train {self.name} departed")
```

```
if __name__ == '__main__':
    conductors = [RailroadsConductor("Kevin"), RailroadsConductor("Melanie")]
    cars = [CoupeCar(conductors[0]), CoupeCar(conductors[1]), EconomyCar(conductors[1]),
EconomyCar(conductors[1]),
```

```
        FreightCar(100, 60, 360, 6 * 360, 1000)]
train = Train("Big Alice", "WHT", cars)
train.depart()

/usr/bin/python3.8 /home/utika/Personal/productivity_stuff/univer/sem_6/OOP/OOP_labs_Univ/
lab_2/main.py
Train Big Alice departed

Process finished with exit code 0
```

Висновок: я ознайомитися з поняттям наслідування в об'єктно-орієнтованому програмуванні, навчитися створювати класи та об'єкти з наслідуванням у Пайтон.